TITLE:

最長共通部分配列計算における run長の対数時間寄与 (計算機科学とアルゴリズムの数理的基礎とその応用)

AUTHOR(S):

酒井, 義文

CITATION:

酒井, 義文. 最長共通部分配列計算におけるrun長の対数時間寄与 (計算機科学とアルゴリズムの数理的基礎とその応用). 数理解析研究所講究録 2011, 1744: 107-114

ISSUE DATE:

2011-06

URL:

http://hdl.handle.net/2433/170965

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
第1744巻 2011年 107-114

最長共通部分配列計算におけるrun長の対数時間寄与

酒井義文*

概要

文字列のrunは，その文字列の同じ文字のみからなる部分文字列である．本稿では，2本の文字列の最長共通部分配列を，一方の文字列のrunの個数と他方の文字列の各runの長さの対数の総和の積の線形時間で求めるアルゴリズムを提案する．

1 はじめに

文字列の類似度を求める問題は，計算機科学，情報検索，分子生物学などの様々な分野において多くの応用がある．二本の文字列の類似度は，通常，両者の長さが一致するように空白記号を挿入するすべての組合せにおいて，対応する位置の文字同士（用いる得点基準によっては，対応する位置の部分文字列同士）の得点の合計の最大値として評価される．動的計画法を用いると，標準的な得点基準のもとで長さ M の文字列 A と長さ N の文字列 B の類似度を $O(MN)$ 時間で求めることができる [15, 7]．特定の符号化のもとで単純な構造をもつ集合から選ばれた部分文字列に分割できる文字列に対しては，より高速に類似度を求めるアルゴリズムが開発されている（たとえば，[14, 6, 8]）．その集合に含まれる部分文字列を擬似的な文字とみなすと，それらのアルゴリズムの多くは，二本の文字列それぞれの任意の位置に現れる擬似的な文字の対に対して，その対に含まれる各擬似的文字が表す部分文字列の長さの和の線形時間の処理を行う．したがって，文字列 A，B がそれぞれ m 個，n 個の擬似的な文字をもつとき，$mN + Mn$ に依存する実行時間がかかる．

run長符号化文字列は，文字列をrunと呼ばれる同じ文字からなる部分文字列に分割し，各部分文字列を，それに現れる共通の文字とそれが並ぶ個数の対に置き換えることで得られる文字列である．文字列に同じ文字が長く連続して現れる場合に，run長符号化文字列はその文字列を簡潔に表すことができるため，この符号化手法は，たとえば画像処理などにおいて，広く用いられている．二本のrun長符号化文字列の類似度を，それらを展開することなしに高速に求めるアルゴリズムが，これまでに数多く提案されている．初めての $O(mN + Mn)$ 時間アルゴリズムは，Bunke と Csirik [5] によって，最長居通部分配列の得点基準もとで動作するものとして与えられた．ただし，m，n はそれぞれ文字列 A，B のrunの個数である．その後，アルゴリズムが動作する得点基準の条件は着実に緩和され，Levenstein距離基準のもとで動作するアルゴリズム [3, 12]，重み付き編集距離基準のもとで動作するアルゴリズム [13]，アフィンギャップペナルティを伴う重み付き編集距離基準のもとで動作するアルゴリズム [9] などが提案されている．

一方，最長共通部分配列基準のもとでは，実行時間が $mN + Mn$ に依存しない，更に高速なアルゴリズムが開発されている．それらのアルゴリズムは二つのグループに分けることができる．一つめのグループは，$O(mn\log(mn))$ 時間で動作するアルゴリズムからなる．Apostolico ら [2] は，run長符号化文字列の間の類似度に関する洗練された漸化式を与え，これにもとづいてアルゴリズムの実行時間を達成した．このグループに含まれるもう一つのアルゴリズムは，Mitchell[11] によるもので，幾何学的な最短路問題への帰着に基づいて設計されている．二つめのグループは，最近になって提案された $O(mN)$ 時間で動作する二つのアルゴリズム [10, 1] からなる．

*東北大学大学院農学研究科

本稿では，最長共通部分配列基準のもとで，Apostolico ら [2] による漸化式にもとづいて $O(m\tilde{n})$ 時間で動作するアルゴリズムを提案する．ただし，$\tilde{n}$ は，A と B がもつ run の長さの対数の総和を表す．このアルゴリズムは，二つめのグループに属する $O(mN)$ 時間アルゴリズムの実行時間を大幅に改善している．また，$0 < c \leq 1$ である定数 c に対して $\Omega(N^c)$ 個の run をもつ文字列 B に対して，このアルゴリズムは，一つめのグループに属するどの $O(mn\log(mn))$ 時間アルゴリズムと比較しても，漸近的に遅く動作することはない．更に，たとえば，文字列 B が定数を超える長さの run を $O(\frac{n}{\log N})$ 個しかもたない場合には，わずか $O(mn)$ 時間で動作する．

2 準備

Σ を有限個の文字からなるアルファベット集合とする．文字列 A の部分配列は，A の必ずしも連続しない任意の位置の文字を任意の個数だけ削除することで得られる文字列である．二本の文字列 A と B に対して，文字列 C が A と B に共通な部分配列で最長のものであるとき，C を A と B の最長共通部分配列とよぶ．一般に，A と B の最長共通部分配列は複数本存在し得る．最長共通部分配列問題は，Σ 上の与えられた任意の文字列 A，B に対して，A と B の最長共通部分配列の任意の一つを求める問題である．

文字列 A，B に対して，AB で，A の後に B が続く連接を表す．任意の文字 a と任意の正整数 M に対して，a^M で，a のみからなる長さ M の文字列を表す．文字列 A の run 長符号化形式は，$A = \alpha(1)^{M(1)}\alpha(2)^{M(2)}\cdots\alpha(m)^{M(m)}$ （ただし，$1 \leq i < m$ である任意の i に対して $\alpha(i) \neq \alpha(i+1)$）であるような m 個の対 $(\alpha(i), M(i))$ の列である．各文字列 $\alpha(i)^{M(i)}$ を，A の第 i 番目の run，あるいは，A の run とよぶ．A の第 i 番目の run を $A(i)$ で表し，連接 $A(g)A(g+1)\cdots A(i)$ を $A(g..i)$ で表す．

本稿では，文字列が run 長符号化形式で与えられる場合の最長共通部分配列問題を考える．以降では，$(\alpha(1), M(1)), \ldots, (\alpha(m), M(m))$，$(\beta(1), N(1)), \ldots, (\beta(n), N(n))$ をそれぞれ A，B の run 長符号化形式とし，$\tilde{n}$ で，1 から n までの各 j に対する $\log_2 N(j)$ の総和を表す．

3 漸化式 [2]

提案するアルゴリズムは以下の漸化式に従って A と B の最長共通部分配列を求める．添え字の対 (i, j) に対して，$L(i, j)$ で，A の接頭辞 $A(1..i)$ と B の接頭辞 $B(1..j)$ の最長共通部分配列の長さを表す．$L(i, 0) = 0$，$L(0, j) = 0$ が成立するのは明らかである．また，$\alpha(i) \neq \beta(j)$ ならば，$L(i, j) = \max(L(i-1, j), L(i, j-1))$ である．このことは，$A(1..i)$ と $B(1..j)$ の最長共通部分配列の末尾の文字が，$\alpha(i)$ と $\beta(j)$ のどちらか少なくとも一方と等しくないことから確認できる．$\alpha(i) = \beta(j)$ の場合は，Apostolico ら [2] による漸化式を用いて $L(i, j)$ を求める．この漸化式は，以下の補題から得られる．

補題 1 ([2]) (i, j) を，$1 \leq i \leq m$，$1 \leq j \leq n$，かつ，$\alpha(i)$ が $\beta(j)$ がどちらも Σ 中の共通の文字 a であるような添え字の対とする．C を，$A(1..i)$ と $B(1..j)$ の最長共通部分配列で，文字 a のみからなる最大の長さの接尾辞 E をもつものとする．D を，$DE = C$ である C の接頭辞とする．g，h をそれぞれ，$A(g..i)$，$B(h..j)$ が E を部分配列としてもつような最大の添え字とする．このとき，D は $A(1..g-1)$ と $B(1..h-1)$ の最長共通部分配列である．

証明 定義より，D の末尾の文字は a ではない．また，$\alpha(g)$，$\beta(h)$ はどちらも a である．よって，$A(1..g-1)$ が D を部分配列としてもたないならば，E は $A(g+1..i)$ の部分文字列でなければならない．このことは，g の定義に矛盾する．$B(1..h-1)$ が D を部分配列としてもつことについても同様に示せる．□

この補題から直ちに以下の漸化式が得られる．ただし，以降では，$M(g..i)$ で，$A(g..i)$ に現れる $\alpha(i)$ と同じ文字の個数を表し，$N(h..j)$ で，$B(h..j)$ に現れる $\beta(j)$ と同じ文字の個数を表す．

系 1 $\alpha(i)$，$\beta(j)$ がどちらも文字 a ならば，$L(i, j)$ は $L(g-1, h-1) + \min(M(g..i), N(h..j))$ の最大値に等しい．ただし，(g, h) の範囲は，$1 \leq g \leq i$，

$1 \leq j \leq h$, かつ, $\alpha(g)$ と $\beta(h)$ のどちらも文字 a であるようなすべての対である.

(g,h) のランクを, $N(1..g) - M(1..h)$ と定義し, $R(g,h)$ で表す. この値を用いることで, 上の漸化式から, 実際のアルゴリズムの中で利用しやすい形の漸化式が得られる.

補題 2 ([2]) (i,j) を, $1 \leq i \leq m$, $1 \leq j \leq n$, かつ, $\alpha(i)$ が $\beta(j)$ がどちらも Σ 中の共通の文字 a であるような添え字の対とする. $X(i,j)$ を, $L(g-1,h-1) - M(1..g')$ の最大値とする. ただし, (g,h) の範囲は, $1 \leq g \leq i$, $1 \leq h \leq j$, $\alpha(g)$ と $\beta(h)$ がどちらも文字 a, かつ, $N(h..j') < M(g..i) \leq N(h..j)$ (すなわち, $R(i,j') < R(g',h') \leq R(i,j)$) である添え字のすべての対である. 同様に, $Y(i,j)$ を, $L(g-1,h-1) - N(1..h')$ の最大値とする. ただし, (g,h) の範囲は, $1 \leq g \leq i$, $1 \leq h \leq j$, $\alpha(g)$ と $\beta(h)$ がどちらも文字 a, かつ, $M(g..i) \leq N(h..j) < M(g..i')$ (すなわち, $R(i,j) \leq R(g',h') < R(i',j)$) である添え字のすべての対である. そのような (g,h) が存在しない場合は, 最大値を $-\infty$ と扱う. この定義において, i' は, $\alpha(i')$ が文字 a であるような i 未満の最大の添え字, あるいは, そのような添え字が存在しない場合は, 0 を表す. 同様に, j' は, $\beta(j')$ が文字 a であるような j 未満の最大の添え字, あるいは, そのような添え字が存在しない場合は, 0 を表す. g', h' も同様に定義される. このとき, $L(i,j) = \max\{L(i,j'), X(i,j) + M(1..i), Y(i,j) + N(1..j), L(i',j)\}$ である.

証明 系 1 より, $M(g..i) \leq N(h..j')$ ならば, $L(i,j') \geq L(g-1,h-1) + M(g..i)$ である. 同様に, $N(h..j) \leq M(g..i')$ ならば, $L(i',j) \geq L(g-1,h-1) + N(h..j)$ である. 一方, $L(i,j')$ と $L(i',j)$ は, どちらも $L(i,j)$ 以下の大きさをもつ. 更に, $M(g..i) = M(1..i) - M(1..g')$, $N(h..j) = N(1..j) - N(1..h')$ である. よって, 系 1 より, 補題は成立する. □

4 アルゴリズム

run 長符号化形式で与えられる二本の文字列の最長共通部分配列を求める $O(m\tilde{n})$ 時間アルゴリズムを提案する. このアルゴリズムは, 先に導入した漸化式に従って $L(i,j)$ を, $(1,1),(1,2),\ldots,(1,n),(2,1),(2,2),\ldots,(m,n)$ の順に計算する. 各添え字 i に対して, アルゴリズムは $L(i,j)$ を以下のように計算する. $\alpha(i) = a$ とする. $\beta(j) \neq a$ のとき, $L(i-1,j)$ と $L(i,j-1)$ の大きい方の値を求めることで, $L(i,j)$ を定数時間で得ることができる. 一方, $\beta(j) = a$ の場合も, もしも $X(i,j)$ と $Y(i,j)$ の値が既に求まっているならば, 補題 2 に従うことで $L(i,j)$ の値を定数時間で得ることができる. 更に, すべての $L(i,j)$ が得られた後には, 標準的なトレースバック処理により, A と B の (run 長符号化形式の) 最長共通部分配列を $O(m+n)$ 時間で求めることができる. よって, 以下では, 各 i に対して, $\beta(j) = \alpha(i)$ であるすべての j に対する $X(i,j)$ と $Y(i,j)$ の求め方についてのみ説明すれば十分である.

なお, アルゴリズムは前処理として, 各 i に対する $M(1..i)$ の値を要素としてもつ配列と, 各 j に対する $N(1..j)$ の値を要素としてもつ配列を構築する. また, その際に, Σ 中の各文字 a に対して, $\alpha(i) = a$ である添え字 i の昇順のリストと, $\beta(j) = a$ である添え字 j の昇順のリストも作成する. これをするのに $O(m+n)$ 時間で十分である.

以下に, Σ 中の各文字 a に関するいくつかの記法を導入する. m_a で, $\alpha(i) = a$ である添え字 i の個数を表し, $i_a(u)$ で, u 番目に小さいそのような添え字 i を表す. 便宜的に, $i_a(0) = 0$ とする. $M(i_a(u))$, $M(i_a(w)..i_a(u))$ をそれぞれ, $M_a(u)$, $M_a(w..u)$ で表す. $\alpha(1) = a$ であるか否かに無関係に, $M_a(1..u)$ と $M(1..i_a(u))$ が同じ値を表すことに注意されたい. 文字列 B に関して n_a, $j_a(v)$, $N_a(v)$, $N_a(x..v)$ も同様に定義する. また, $\tilde{n}_a$ で, 1 から n_a までの各添え字 v に対する $\log_2 N_a(v)$ の総和を表す. $R_a(u,v)$, $X_a(u,v)$, $Y_a(u,v)$ でそれぞれ, $R(i_a(u),j_a(v))$, $X(i_a(u),j_a(v))$, $Y(i_a(u),j_a(v))$ を表す. 更に, $r_a(u,v)$ を, $0 \leq w \leq m_a$ かつ $0 \leq x \leq n_a$ であるすべての対 (u,v) に対するランク $R_a(w,x)$ の中で, $R_a(u,v)$ が $r_a(u,v)$ 番目に小さいランクであるような順位とする. $X_{a,u}(r)$ を,

$L(i_a(w)-1, j_a(x)-1) - M_a(1..w-1)$ の最大値とする．ただし，(w,x) の範囲は，$1 \leq w \leq u$，$1 \leq x \leq n_a$，かつ，$r_a(w-1,x-1) = r$ である添え字のすべての対とする．同様に，$Y_{a,v}(r)$ を，$L(i_a(w)-1, j_a(x)-1) - N_a(1..x-1)$ の最大値とする．ただし，(w,x) の範囲は，$X_{a,v}(r)$ の場合と同様である．そのような (w,x) が存在しない場合は，最大値を $-\infty$ と扱う．$w \leq u$ かつ $v < x$ のとき $r_a(u-1,v) \leq r_a(w-1,x-1)$ であるから，$X_a(u,v)$ は，$X_{a,u}(r)$ の最大値として求まる．ただし，r の範囲は，$r_a(u,v-1) < r \leq r_a(u,v)$ であるすべての値である．同様に，$Y_a(u,v)$ は，$Y_{a,v}(r)$ の最大値として求まる．ただし，r の範囲は，$r_a(u,v) \leq r < r_a(u-1,v)$ であるすべての値である．

次の節において，すべての対 (u,v) に対する $r_a(u,v)$ を $O(m_a\tilde{n}_a)$ 時間で求める方法を与える．その次の二つの節では，すべての添え字 v に対する $X_a(u,v)$ と $Y_a(u,v)$ を $O(\tilde{n}_a)$ 時間で求める方法をそれぞれ与える．Σ 中の各文字 a に対する $m_a\tilde{n}_a$ の総和は高々$m\tilde{n}$ であるから，これらにより，$O(m\tilde{n})$ 時間で A と B の最長共通部分配列を求めるアルゴリズムが得られることになる．

4.1 $r_a(u,v)$ を求める方法

$0 \leq u \leq m_a$ かつ $0 \leq v \leq n_a$ であるすべての対 (u,v) に対する $r_a(u,v)$ の値を $O(m_a\tilde{n}_a)$ 時間で求める方法について述べる．それらの値を求めるには，すべての対 (u,v) が $R_a(u,v)$ の値の昇順に並んだリストを作成すればいよい．なぜなら，そのリストに現れる対を先頭から末尾まで順にたどることで，線形時間ですべての対 (u,v) に対する $r_a(u,v)$ の値を求めることができるからである．

t を，2^t が $N_a(1..n_a)$ より大きい最小の自然数とする．$0 \leq s \leq t$ である任意の添え字 s に対して，$2^s l$ から $2^s(l+1)-1$ までの範囲を，(s,l) 範囲とよぶことにする．すると，$R_a(u,v) = N_a(1..v) - M_a(1..u)$ より，$M_a(1..u)$ が (s,i) 範囲にあり，かつ，$N_a(1..v)$ が (s,j) 範囲にあるような任意の対 (u,v) に対して，$2^s((j-i)-1) < R_a(u,v) < 2^s((j-i)+1)$ が成立する．I^s を，添え字 u が存在して，$M_a(1..u)$ が (s,l) 範囲にあるようなすべての添え字 l の集合とし，同様に，J^s を，添え字 v が存在して，$N_a(1..v)$ が (s,l) 範囲にあるようなすべての添え字 l の集合とする．更に，K^s を，$I^s \times J^s$ 中のすべての対 (i,j) が $j-i$ の昇順にならんだリストとする．したがって，もしも K^0 中の各対 $(M_a(1..u), N_a(1..v))$ に対する (u,v) が定数時間で求められるならば，K^0 をすべての対 (u,v) が $R_a(u,v)$ の値の昇順に並んだリストとして扱うことができる．このことに基づいて，リスト K^t を作成した後に，K^s を K^{s-1} に次々と更新することによって K^0 を作成する．

はじめに，I^s と J^s を，要素が昇順に並ぶリストとして作成する方法を以下に述べる．定義より，I^0 は，0から m_a までのすべての添え字 u に対する $M_a(1..u)$ の値の昇順のリストである．このリストは，アルゴリズムの前処理で作成した $M(1..i)$ を値としてもつ配列と，$\alpha(i) = a$ である添え字 i の昇順のリストから，線形時間で作成できる．これを作成する際に，I^s の各要素 $M_a(1..u)$ に添え字 u を添付しておく．これにより，K^0 中の各対 $(M_a(1..u), N_a(1..v))$ に対する u を定数時間で得ることができるようになる．$s \geq 1$ である I^s は，以下のようにすることで，I^{s-1} から線形時間で作成できる．すなわち，I^{s-1} の要素 l を先頭から順に読み込み，$\lfloor \frac{l}{2} \rfloor$ を I^s の要素とすればよい．ただし，l が偶数で，$l+1$ も I^{s-1} の要素である場合は，I^s に同じ値が要素として重複して含まれることのないようにする必要があるため，$\lfloor \frac{l+1}{2} \rfloor$ を要素として加えない．また，後に K^s を作成する際の便宜のために，I^{s-1} 中の各要素 l へのポインタを，I^s の要素 $\lfloor \frac{l}{2} \rfloor$ にもたせる．これにより，I^s 中の任意の要素 l' から I^{s-1} 中の要素 $2l'$，$2l'+1$ に，もしそれらが存在するならば，定数時間でアクセスすることができるようになる．J^s も同様に作成する．

次に，上で作成した I^s，J^s を用いて K^{s-1} を K^s から作成する方法について述べる．t の定義より，J^t は要素0のみをもつ．したがって，K^t は，I^t に属するすべての添え字 k に対する対 $(k,0)$ からなる．これを作成するのに $O(|K^t|)$ 時間で十分である．K^s の各要素 (i,j) は，I^s 中の要素 i へのポインタと，J^s 中の要素 j へのポインタをもつと仮定する．このと

き，K^{s-1} は以下のように作成される．まず，空のリスト K を用意する．次に，以下を繰り返す．K^s の先頭から順に，$j-i$ が同じ値をとる対 (i,j) 全体からなる部分リストを取り出す．この部分リスト中の各対 (i,j) に対して，I^{s-1} が $2i+1$ をもち，かつ，J^{s-1} が $2j$ をもつならば，$(2i+1,2j)$ を K の末尾に要素として加える．その際に，$(2i+1,2j)$ に I^{s-1} 中の $2i+1$ と J^{s-1} 中の $2j$ へのポインタをもたせる．次に，K^s の同じ部分リスト中の各対 (i,j) に対して，I^{s-1} が $2i$ をもち，かつ，J^{s-1} が $2j$ をもつならば，$(2i,2j)$ を K の末尾に加える．同様に，I^{s-1} が $2i+1$ をもち，かつ，J^{s-1} が $2j+1$ をもつならば，$(2i+1,2j+1)$ を加える．さらに，次に，K^s の同じ部分リスト中の各対 (i,j) に対して，I^{s-1} が $2i$ をもち，かつ，J^{s-1} が $2j+$ をもつならば，$(2i,2j+1)$ を K の末尾に加える．それらの対を K に加える際に，I^{s-1}，J^{s-1} 中の要素へのポインタをもたせるのは，$(2i+1,2j)$ の場合と同様である．以上を K^s が空になるまで繰り返せば，K^{s-1} が K の値として得られる．K^s，I^s，J^s の要素がもつポインタにより，K^{s-1} の各対は定数時間で求まるから，K^{s-1} を求めるのに，それに含まれる要素数，すなわち，$|I^{s-1}|\times|J^{s-1}|$ の線形時間で十分である．

以上の議論により，$0\le u\le m_a$，$0\le v\le n_a$ であるすべての対 (u,v) に対する $r_a(u,v)$ を $O(m_a\tilde{n}_a)$ 時間で求めることができることを証明するためには，$0\le s\le t$ である任意の s に対して $|I^s|\le m_a+1$，かつ，$0\le s\le t$ であるすべての s に対する $|J^s|$ の総和が $O(\tilde{n}_a)$ であることを示せば十分である．$|I^0|=m_a+1$，かつ，$1\le s\le t$ である任意の s に対して $|I^s|\le|I^{s-1}|$ であるから，以下では，$0\le s\le t$ であるすべての s に対する $|J^s|$ の総和が $O(\tilde{n}_a)$ であることを示す．

$0\le s\le t$ である s と J^s に属する l からなる対 (s,l) を四つのグループに分類する．第1のグループは，J^{s-1} が定義されていない，すなわち，J^0 に属する l に対する $(0,l)$ からなる．第2のグループは，J^{s-1} に $2l$ と $2l+1$ がどちらも属する (s,l) からなる．第3のグループは，J^{s-1} に $2l+1$ は属するが，$2l$ は属さない (s,l) からなる．第4のグループは，J^{s-1} に $2l$ は属するが，$2l+1$ は属さない (s,l) からなる．したがって，これら四つのグループに属する対の総数が $O(\tilde{n}_a)$ であることを示せば十分である．

第1のグループには，J^0 に属する l と同数，すなわち，n_a+1 個の (s,l) が属する．一方，J^{s-1} の長さはちょうど J^s から第2のグループに (s,l) が属する l をすべて取り除いた長さに等しい．したがって，$|J^0|=n_a+1$，$|J^t|=1$ より，第2のグループに属する対の個数はちょうど n_a 個である．以下に，第3のグループに属する対の個数を見積もる．第3のグループに属する任意の対 (s,l) に対して，$v(s,l)$ で，$N_a(1..v)$ が (s,l) 範囲のある最小の添え字 v を表すことにする．したがって，第3のグループの定義より，$N_a(1..v(s,l))$ は $(s-1,2l+1)$ 範囲にあり，一方，$N_a(1..v)$ が $(s-1,2l)$ 範囲にある v は存在しない．このことから，$v(s,l)=v(s',l')$ であるような第3のグループに属する互いに異なる対 (s,l) と (s,l') に対して，$s\neq s'$ でなければならないため，$v(s,l)$ の値が等しい対 (s,l) の個数は，たかだか $\log_2(N_a(1..v)-N_a(1..v-1))=\log_2 N_a(v)$ である．よって，第3のグループに属する対の個数は，たかだか $\tilde{n}_a$ である．同様の議論により，第4のグループに属する対の個数は，たかだか $2\tilde{n}_a+1$ 個である．第3のグループの対の個数との差異は，$N_a(1..0)=0$ であるのに対して，$N_a(1..n_a)$ が 2^t よりもかなり小さな値をとり得ることによる．以上より，$0\le s\le t$ であるすべての s に対する $|J^s|$ の総和は $O(\tilde{n}_a)$ であり，したがって，$0\le u\le m_a$，$0\le v\le n_a$ であるすべての対 (u,v) に対する $r_a(u,v)$ の値を求めるのに $O(m_a\tilde{n}_a)$ 時間で十分である．

4.2 $X_a(u,v)$ を求める方法

本節では，$1\le u\le m_a$ である任意の u に対して，$1\le v\le n_a$ であるすべての v に対する $X_a(u,v)$ の値を $O(\tilde{n}_a)$ 時間で求める方法について説明する．

アルゴリズムは，2^t 個の葉をもつ完全二分木 S_a として表されるデータ構造を利用する．ただし，t は，2^t が $r_a(0,n_a)$ 以上である最小の自然数である．この設定により，任意の対 (u,v) に対する $r_a(u,v)$ に対応する S_a の葉が，左から第 $r_a(u,v)$ 番目の葉

として存在する．なぜなら，$R_a(u,v) = N_a(1..v) - M_a(1..u) \leq N_a(1..n_a) - M_a(1..0) = R_a(0,n_a)$ より，$r_a(u,v) \leq r_a(0,n_a)$ であるからである．根を除けば，S_a のある高さにおける左から第 l 番目の頂点の親は，一つ上の高さにおける左から第 $\lceil \frac{l}{2} \rceil$ 番目の頂点である．S_a のある高さにある左から第 l 番目の頂点の右隣は，同じ高さの左から第 $l+1$ 番目の頂点である．S_a の任意の頂点は，もしも右隣が存在するならば，それへのリンクをもつものとする．S_a のある葉が左から r 番目のものであるとき，位置 r にあるという．

u に対して，すべての v に対する $X_a(u,v)$ を求める際に，S_a の位置 r の葉が $X_{a,u}(r)$ の値をもつとき，この葉は有効であるという．また，S_a の任意の内部頂点に対して，それがもつ二つの子が再帰的に有効であり，かつ，それらの子の最大値を自身がもつとき，その頂点は有効であると定義する．したがって，有効である任意の頂点は，子孫であるすべての葉における最大値をもつ．無効な頂点は，もはや有効ではない可能性のある頂点をいう．アルゴリズムは，$1 \leq v \leq n_a$ である各 v に対する位置 $r_a(u,v-1)+1$，$r_a(u,v)$ の間に子孫である葉がすべてあるような無効な頂点のみを有効になるように更新する．なぜなら，その範囲にある葉の最大値が $X_a(u,v)$ であるからである．この方針のもとで，アルゴリズムを以下の補題を用いて設計する．以降では，子孫である葉がすべて位置 p，q の間の範囲にある頂点を，p と q の間の範囲に関連する頂点という．また，この範囲に関連するすべての頂点が有効であるとき，この範囲は有効であるという．

補題 3 $0 \leq q_0 < q_1 < \cdots < q_f \leq r_a(0,n_a)$ である任意の $q_0, q_1, \ldots, q_f$ に対して，もしも $1 \leq e \leq f$ である任意の e に対して $q_{e-1}+1$ と q_e の範囲が有効ならば，$O(\sum_{1 \leq e \leq f} \log(q_e - q_{e-1}))$ 時間で，位置 q_0+1，q_f の間の範囲にある葉の最大値を求め，かつ，この範囲を有効にすることができる．

証明 p と q の間の範囲に関連する頂点で，その親がもはやその範囲に関連しないものを，支配的な頂点ということにする．位置 p，q の間の範囲にある任意の葉に対して，これを子孫とするこの範囲に関連するたった一つの支配的な頂点が存在することに注意されたい．したがって，もしも p と q の間が有効ならば，この範囲に関連するすべての支配的な頂点の最大値を求めることで，この範囲にあるすべての葉の最大値が得られる．一方の頂点の子孫である葉の最も左のものが他方の頂点の子孫である葉の最も右のものの右隣であるとき，二つの頂点は隣接するという．更に，位置 p の葉が最初の頂点の子孫である葉の最も左のものであり，すべての連続する二つの頂点が隣接するような位置 p，q の間の範囲に関連するすべての支配的な頂点からなるリストを，標準的であるという．このリストの末尾の頂点の子孫である葉の最も右のものは，位置 q にある．標準的なリスト中の任意の連続する三つの頂点に対して，2 番目の頂点の子孫である葉の個数は，他の二つの頂点の少なくとも一方の子孫である葉の個数のたかだか 2 倍である．さもなければ，支配的ではないからである．このことは，p と q の間の範囲に関連する支配的な頂点の個数がたかだか $2\log_2(q-p)$ であることを意味する．以下の手続きを実行することで，$O(\log(q-p))$ で p と q の間の範囲に関連する支配的な頂点の標準的なリストを求められることを確認することは難しくない．

1. 親，親の右隣，右隣，右側の子の右隣，右側の子の優先順で，p と q の間の範囲に関連する頂点を次々と訪れることで，S_a 上の経路を位置 p にある葉から位置 a にある葉へとたどり，
2. その際に訪れた頂点で直前や直後の頂点の子ではないものすべてを，訪れた順に並べたリストとして返す．

以上に基づいて，補題を証明する．位置 q_0+1，q_f の間の範囲にある葉の最大値は，1 から f までの各 e に対する $q_{e-1}+1$ と q_e の間の範囲に関連する支配的な頂点の標準的なリストの e の昇順の連接を作成し，得られたリストに現れる頂点の最大値を求めることで，$O(\sum_{1 \leq e \leq f} \log(q_e - q_{e-1}))$ 時間で得ることができる．得られたリストは，$q_{e-1}+1$ と q_e の間の範囲を有効にする際に，有効にする頂点を見つけることにも利用する．このリストを最初の頂点から

最後の頂点へとたどり，訪れた各頂点に対して，もしも直前の頂点と共通の親をもつならば，両者の最大値を親の値とすることで有効化し，リスト中のその二つ頂点を共通の親に置き換える操作を繰り返す．これにより，リストは最終的に q_0+1 と q_f の間の範囲に関連する支配的な頂点の標準的なリストに変わり，この範囲の有効化が完了する．これを行うのに，元のリストの長さの線形時間で十分である．□

以下に，アルゴリズムの動作を説明する．初期設定として，S_a 中のすべての頂点の値を $-\infty$ とする．また，各 u に対して，$1 \leq v \leq n_a$ であるすべての v に対する $X_a(u,v)$ を求める際に，$r_a(u-1,v-1)+1$ と $r_a(u-1,v)$ の間の範囲を有効にする．この仮定により，u に対して，$1 \leq v \leq n_a$ であるすべての v に対する $X_a(u,v)$ を求める作業の開始時において，$0 \leq v \leq n_a$ である $r_a(u-1,v-1)+1$ と $r_a(u-1,v)$ の間の範囲に関連する頂点で，$1 \leq v \leq n_a$ であるどの v に対する位置 $r_a(u-1,v-1)$ にある葉も子孫としてもたないものはすべて有効である．ただし，便宜的に $r_a(u-1,-1)=1$ と扱うことにする．$P=p_1,p_2,\ldots,p_z$ を，$0 \leq v \leq n_a$ であるすべての v に対する $r_a(u-1,v)$ と $r_a(u,v)$ からなる昇順のリストとする．$p_{y-1}=r_a(u,n_a)$ かつ $p_y=r_a(u-1,0)$ ならば，$p_y=p_{y-1}+1$ であるから，$\sum_{1<y\leq z}\log(p_y-p_{y-1})=O(\tilde{n}_a)$ であることに注意されたい．

アルゴリズムは，最初に，$1 \leq y \leq z$ であるすべての y に対する $p_{y-1}-1$ と p_y の間の範囲を有効にする．これをするのに，$1 \leq v \leq n_a$ である各 v に対して，位置 $r_a(u-1,v-1)$ にある葉の値を，$L(i_a(u)-1,j_a(v)-1)-M_a(1..u-1)$ がそれまでの値よりも大きいならば，これに更新し，$r_a(u-1,v-1)=p_y$ のとき，位置 $r_a(u-1,v-1)$ にある葉からの $p_{y-1}-1$ と p_y の間の範囲に関連する頂点を根へとたどりながら，訪れた頂点の二つの子の最大値を自身の値とすればよい．その際に訪れる頂点の個数は，たかだか $\log_2(p_y-p_{y-1})$ であり，よって，これの線形時間で $p_{y-1}-1$ と p_y の間の範囲を有効化できる．したがって，$O(\tilde{n}_a)$ 時間で $1 \leq y \leq z$ であるすべての y に対する $p_{y-1}-1$ と p_y の間の範囲を有効にすることができる．

次に，アルゴリズムは，$1 \leq v \leq n_a$ である各 v に対して，位置 $r_a(u,v-1)-1$，$r_a(u,v)$ の間の範囲にある葉の最大値，すなわち $X_a(u,v)$ を求め，更に，この範囲を有効化する．補題3を用いれば，すべての v に対してこれをするのに $O(\sum_{1<y\leq z}\log(p_y-p_{y-1}))=O(\tilde{n}_a)$ 時間で十分である．

以上より，上に述べたアルゴリズムは，$O(m_a n_a)$ 時間で S_a の各頂点の値を $-\infty$ に初期設定した後に，昇順に1から m_a までの各 u に対して，$O(\tilde{n}_a)$ 時間で，$1 \leq v \leq n_a$ であるすべての v に対する $X_a(u,v)$ の値を求め，$r_a(u,v-1)+1$ と $r_z(u,v)$ の間の範囲を有効化する．

4.3 $Y_a(u,v)$ を求める方法

昇順に1から m_a までの各 u に対して，$1 \leq v \leq n_a$ であるすべての v に対する $Y_a(u,v)$ を $O(\tilde{n}_a)$ 時間で求める方法について述べる．$X_a(u,v)$ を求める再に用いた S_a と同様に定義される木 T_a を用いる．ただし，位置 r にある葉が有効であるための条件を，その葉の値が $Y_{a,u}(r)$ であることとする．

$X_a(u,v)$ を求める場合には，$r_a(u,v-1)-1$ と $r_a(u,v)$ の間の範囲にたかだか $N_a(v)$ 個の葉しか存在しないことを利用した．しかし，$Y_a(u,v)$ を求める場合には，$r_a(u,v)$ と $r_a(u-1,v)-1$ の間の範囲にはたかだか $M_a(u)$ 個の葉しか存在しないことはいえても，一般に，この個数は $N_a(v)$ の値を超えることがあるため，$X_a(u,v)$ の場合と同じ方法をそのまま利用してもうまくいかない．そこで，各 u に対して，すべての v に対する $Y_a(u,v)$ を求め，$r_a(u,v)$ と $r_a(u-1,v)-1$ の間の範囲を有効化する代わりに，$1 \leq v' < n'_a$ であるすべての v' に対する位置 $r'_a(u,v')$，$r'_a(u,v'+1)-1$ の間の範囲にある葉の最大値を $Y'_a(u,v')$ として求め，この範囲を有効化する．ただし，$r'_a(u,v')$ は，$1 \leq v \leq n_a$ であるすべての v に対する $r_a(u-1,v)$ と $r_a(u,v)$ の中で第 v' 番目に小さな値を表し，$r'_a(u,n'_a)=r_a(u-1,n_a)$ である．これをするのに，$X_a(u,v)$ と同様の方法を用いることで，$O(\tilde{n}_a)$ 時間で十分である．なぜなら，$r'_a(u,v')=r_a(u,n_a)$ かつ $r'_a(u,v'+1)=r_a(u-1,1)$

ならば，$r'_a(u,v'+1)-r'_a(u,v') \leq N_a(1)+1$ であるからである．

得られた $Y'_a(u,v')$ の値から $Y_a(u,v)$ を求めるために，Ann ら [1] の $O(mN)$ 時間アルゴリズムにならい，線形時間の初期化の後に任意に指定した範囲の最大値を定数時間で得ることのできるデータ構造 [4] を利用する．このデータ構造は，すべての v' に対する $Y'_a(u,v')$ に関する初期化を $n'_a \leq 2n_a$ の線形時間で行った後に，与えられた任意の c，d に対して，$Y'_a(u,c), Y'_a(u,c+1), \ldots, Y'_a(u,d)$ の最大値を返す．これを用いることによって，アルゴリズムは $1 \leq v \leq n_a$ であるすべての v に対する $Y_a(u,v)$ を $O(n_a)$ 時間で求めることができる．

5 まとめ

長さ M，run の個数 m の文字列 A と長さ N，run の個数 n の文字列 B が run 長符号化形式で与えられたときに，A と B の最長共通部分配列を求める $O(m\tilde{n})$ 時間アルゴリズムを提案した．ただし，$\tilde{n}$ は，B の各 run の長さの対数の総和である．このアルゴリズムは，B が定数を超える長さの run を $O(\frac{n}{\log N})$ しかもたないとき，$O(mn)$ 時間で動作する．しかし，一般の場合に $O(mn)$ 時間で動作するアルゴリズムが存在するか否かは判っていない．また，たとえば重み付き編集距離基準などのより制限の緩い得点基準のもとで，最長共通部分配列の場合と同じ $O(mn\log(mn))$ 時間や $O(m\tilde{n})$ 時間で二本の文字列の類似度を求めることができるか否かも，未解決な問題の一つである．

参考文献

[1] H.Y. Ann, C.B. Yang, C.T. Tseng, C.Y. Hor, A fast and simple algorithm for computing the longest common subsequence of run-length encoded strings, Information Processing Letters, 108 (2008) 360-364.

[2] A. Apostolico, G.M. Landau, S. Skiena, Matching for run-length encoded strings, Journal of complexity, 15 (1999) 4–16.

[3] O. Arbell, G.M. Landau, J.S.B. Mitchell, Edit distance of run-length encoded strings, Information Processing Letters, 83 (2002) 307–314.

[4] M.A. Bender, M. Farach-Colton, The LCA problem revisited, in: LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, 2000, pp. 88–94.

[5] H. Bunke, J. Csirik, An improved algorithm for computing the edit distance of run-length coded strings, Information processing letters, 54 (1995) 93–96.

[6] M. Crochemore, G.M. Landau, M. Ziv-Ukelson, A subquadratic sequence alignment algorithm for unrestricted scoring matrices, SIAM Journal on Computing, 32 (2003) 1654–1673.

[7] O. Gotoh, An improved algorithm for matching biological sequences, Journal of Molecular Biology, 162 (1982) 705–708.

[8] D. Hermelin, G.M. Landau, S. Landau, O. Weimann, A unified algorithm for accelerating edit-distance computation via text-compression, Symposium on Theoretical Aspect of Computer Science 2009 (Freiburg), pp. 529–540.

[9] J.W. Kim, A. Amir, G.M. Landau, K. Park, Computing similarity of run-length encoded strings with affine gap penalty, Theoretical Computer Science, 395 (2008) 268–282.

[10] J.J. Liu, Y.L. Wang, R.C.T. Lee, Finding a longest common subsequence between a run-length-encoded string and an uncompressed string, Journal of complexity, 24 (2008) 173–184.

[11] J. Mitchell, A geometric shortest path problem, with application to computing a longest common subsequence in run-length encoded strings, Technical Report, Department of Applied Mathematics, SUNY Stony Brook, NY, 1997.

[12] V. Mäkinen, G. Navarro, E. Ukkonen, Approximate Matching of run-length compressed strings, in : 12th Annual Symposium on Combinatorial Pattern Matching, in: Lecture Notes in Computer Science, vol. 2089, 2001, pp. 31–49.

[13] V. Mäkinen, G. Navarro, E. Ukkonen, Approximate Matching of run-length compressed strings, Algorithmica, 35 (2003) 347–369.

[14] W.J. Masek, M.S. Paterson, A fast algorithm for computing string-edit distances, Journal of Computer and System Sciences, 20 (1980) 18–31.

[15] R.A. Wagner, M.J. Fischer, The string to string correction problem, Journal of the Association for Computing Machinery, 21 (1974) 168–173.